

Общи задачи

Тодор Дуков

Примери за задачи с числа

Искаме да напишем алгоритъм, който при подадена двойка от естествени числа $\langle a, b \rangle$, където $b \leq a$, да върне $\text{gcd}(a, b)$ т.е. това число $d \in \mathbb{N}$, за което:

- $d \mid a$ и $d \mid b$
- ако $d_1 \mid a$ и $d_1 \mid b$, то $d_1 \mid d$

За целта ще покажем, че за всяко $a, b, q, r \in \mathbb{N}$, където $0 < b \leq a$, $a = bq + r$ и $r \in \{0, \dots, b-1\}$, е изпълнено, че:

$$\text{gcd}(a, b) = \text{gcd}(b, r)$$

Първо имаме, че $\text{gcd}(a, b) \mid a$ и $\text{gcd}(a, b) \mid b$, откъдето понеже $a = bq + r$, имаме $\text{gcd}(a, b) \mid r$. Тогава $\text{gcd}(a, b) \mid \text{gcd}(b, r)$. От друга страна $\text{gcd}(b, r) \mid b$ и $\text{gcd}(b, r) \mid r$, откъдето $\text{gcd}(b, r) \mid bq + r = a$. Така $\text{gcd}(b, r) \mid \text{gcd}(a, b)$. Накрая можем да заключим $\text{gcd}(a, b) = \text{gcd}(b, r)$ от $\text{gcd}(a, b) \mid \text{gcd}(b, r)$ и $\text{gcd}(b, r) \mid \text{gcd}(a, b)$.

На базата на това наблюдение се получава следният алгоритъм (кръстен на Евклид):

```
1 int euclid(int a, int b) // a >= b
2 {
3     if (b == 0)
4         return a;
5
6     return euclid(b, a % b);
7 }
```

Ще покажем с индукция относно b , че за всяко $a \geq b$, е изпълнено, че:

$$\text{euclid}(a, b) = \text{gcd}(a, b)$$

- В базовия случай имаме $\text{euclid}(a, 0) = a = \text{gcd}(a, 0)$
- Нека $b > 0$ и $a = bq + r$ за някои $r \in \{0, \dots, b-1\}$ и q и нека твърдението е изпълнено за всяко $b' < b$. Тогава:

$$\text{euclid}(a, b) = \text{euclid}(b, r) \stackrel{(\text{ИП})}{=} \text{gcd}(b, r) = \text{gcd}(a, b)$$

Алгоритъмът терминира, понеже управляващият параметър b винаги намаля, докато не стане 0. На пръсти ще покажем, че алгоритъмът има сложност по време и памет $O(\log(a))$. Нека видим, че ако $a > b$, то $\text{mod}(a, b) < \frac{a}{2}$:

1 сл. ако $b \leq \frac{a}{2}$, то $\text{mod}(a, b) < \frac{a}{2}$ и сме готови

2 сл. ако пък $b > \frac{a}{2}$, то тогава $a - b < \frac{a}{2}$, откъдето $\text{mod}(a, b) = \text{mod}(a - b, b) = a - b < \frac{a}{2}$

Тогава през на всеки две стъпки на алгоритъма от вход $\langle a, b \rangle$, отиваме до вход $\langle \text{mod}(a, b), \text{mod}(b, \text{mod}(a, b)) \rangle$ и левият аргумент става поне два пъти по-малък. Тъй като левият аргумент е горна граница за десния, то той също ще намаля рязко. Дълбочината на дървото на рекурсията зависи само от броя на рекурсивните извиквания. Понеже те са логаритмично много и всяко едно от тях заема константна памет, сложността по памет е също $O(\log(a))$.

Втората задача за числа е скрита в задача за масиви:

Имаме един масив $A[1 \dots n]$, който съдържа всички числа от 1 до n , с изключение на едно от тях, което е заместено с друго число от 1 до n . Искаме да напишем колкото се може по-бърз алгоритъм, който при вход такъв масив A връща наредената двойка $\langle \text{dup}, \text{miss} \rangle$ от дубликата и липсващото число. Оказва се, че тази задача може да се реши за линейно време и константна памет. За целта трябва да се забележат следните факти:

- $\text{dup} - \text{miss} = \sum_{i=1}^n A[i] - \sum_{i=1}^n i = \sum_{i=1}^n A[i] - \frac{n(n+1)}{2} =: S_{1,A}$
- $(\text{dup} + \text{miss})(\text{dup} - \text{miss}) = \text{dup}^2 - \text{miss}^2 = \sum_{i=1}^n A[i]^2 - \sum_{i=1}^n i^2 = \sum_{i=1}^n A[i]^2 - \frac{n(n+1)(2n+1)}{6} =: S_{2,A}$

От тях получаваме следната система от уравнения:

$$\begin{cases} \text{dup} - \text{miss} = S_{1,A} \\ \text{dup} + \text{miss} = \frac{S_{2,A}}{S_{1,A}} \end{cases}$$

Най-сложното, което трябва да направим, е да пресметнем $S_{1,A}$ и $S_{2,A}$:

```

1 pair<int, int> find_missing_and_duplicate(int *arr, int n)
2 {
3     int sum = (n * (n + 1)) / 2;
4     int sq_sum = (n * (n + 1) * (2 * n + 1)) / 6;
5
6     int arr_sum = 0;
7     int arr_sq_sum = 0;
8
9     for (int i = 0; i < n; ++i)
10    {
11        arr_sum += arr[i];
12        arr_sq_sum += arr[i] * arr[i];
13    }
14
15    int dup = ((arr_sum - sum) + (arr_sq_sum - sq_sum) / (arr_sum - sum)) / 2; // (S1 + S2 / S1) / 2
16    int miss = dup - arr_sum + sum; // dup - S1
17
18    return {dup, miss};
19 }
```

Остава само да се докаже следното твърдение:

Инвариант. При всяко достигане на проверката за край на цикъла на ред 9 имаме, че:

$$\text{arr_sum} = \sum_{k=0}^{i-1} \text{arr}[k] \text{ и } \text{arr_sq_sum} = \sum_{k=0}^{i-1} \text{arr}[k]^2$$

База. При първото достигане имаме, че:

- $\text{arr_sum} = 0 = \sum_{k=0}^{0-1} \text{arr}[k]$
- $\text{arr_sq_sum} = 0 = \sum_{k=0}^{0-1} \text{arr}[k]^2$

Поддръжка. Нека твърдението е изпълнено за някое непоследно достигане на проверката за край на цикъла. Тогава влизайки в тялото на цикъла:

- arr_sum става $\text{arr_sum} + \text{arr}[i] \stackrel{(\text{ИП})}{=} \sum_{k=0}^{i-1} \text{arr}[k] + \text{arr}[i] = \sum_{k=0}^i \text{arr}[k] = \sum_{k=0}^{i_{\text{new}}-1} \text{arr}[k]$
- arr_sq_sum става $\text{arr_sq_sum} + \text{arr}[i]^2 \stackrel{(\text{ИП})}{=} \sum_{k=0}^{i-1} \text{arr}[k]^2 + \text{arr}[i]^2 = \sum_{k=0}^i \text{arr}[k]^2 = \sum_{k=0}^{i_{\text{new}}-1} \text{arr}[k]^2$

Терминация. В последното достигане на проверката за край на цикъла имаме, че $i = n$, откъдето:

$$\text{arr_sum} = \sum_{i=0}^{n-1} \text{arr}[i] \text{ и } \text{arr_sq_sum} = \sum_{i=0}^{n-1} \text{arr}[i]^2$$

Имайки това твърдение, остава само да се отбележи, че dup и miss наистина са решения на горната система от уравнения (със съответните преиндексирания). Накрая нека за пълнота с индукция по $n \in \mathbb{N}$ покажем, че:

$$\sum_{i=0}^n i = \frac{n(n+1)}{2} \text{ и } \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

- $\sum_{i=0}^0 i = 0 = \frac{0(0+1)}{2}$ и $\sum_{i=0}^0 i^2 = \frac{0(0+1)(2 \cdot 0+1)}{6}$ ✓
- $\sum_{i=0}^{n+1} i = \sum_{i=0}^n i + (n+1) \stackrel{(\text{ИП})}{=} \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$
- $\sum_{i=0}^{n+1} i^2 = \sum_{i=0}^n i^2 + (n+1)^2 \stackrel{(\text{ИП})}{=} \frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{n(n+1)(2n+1)}{6} + \frac{6(n+1)^2}{6} = \frac{(n+1)(2n^2+n+6n+6)}{6} = \frac{(n+1)(n+2)(2(n+1)+1)}{6}$

Задачи

Задача 1. Да се напише алгоритъм `find_primes(n)`, който приема естествено число n и връща булев масив $P[2 \dots n]$, за който е изпълнено, че за всяко $2 \leq i \leq n$:

$$P[i] \text{ е истина } \iff i \text{ е просто}$$

След това да се докаже неговата коректност и да се изследва сложността му по време и памет.

Задача 2. Масив на Монж ще наричаме всеки двумерен масив $A[1 \dots n, 1 \dots m]$ от естествени числа, за който:

$$A[p, q] + A[s, t] \leq A[p, t] + A[s, q] \text{ за всички } 1 \leq p, s \leq n \text{ и } 1 \leq q, t \leq m$$

Да се напише алгоритъм `test_mongeness(A)` със линейна сложност, който приема двумерен масив $A[1 \dots n, 1 \dots m]$ и проверява дали е масив на Монж. След това да се докаже неговата коректност и да се изследва сложността му по време и памет.

Задача 3. Да се напише колкото се може по-бърз алгоритъм `split(A)`, който приема масив $A[1 \dots n]$ от цели числа и го подрежда така, че всички отрицателни числа да са вляво от всички неотрицателни. След това да се докаже неговата коректност и да се изследва сложността му по време и памет.

Задача 4. Да се напише колкото се може по-бърз алгоритъм `find_products(A)`, който приема масив $A[1 \dots n]$ от цели числа и връща масив $B[1 \dots n]$, такъв че за всяко $1 \leq i \leq n$:

$$B[i] = \prod_{\substack{k=1 \\ k \neq i}}^n A[k]$$

След това да се докаже неговата коректност и да се изследва сложността му по време и памет.

Задача 5. Да се напише колкото се може по-бърз алгоритъм `find_max_product_pair(A)`, който приема масив $A[1 \dots n]$ от цели числа и връща двойка $\langle i, j \rangle$ от различни индекси, за които $A[i] * A[j]$ е максимално. След това да се докаже неговата коректност и да се изследва сложността му по време и памет.

Задача 6. Даден е следният алгоритъм:

```
1  int alg(int *arr, int n)
2  {
3      bool x = true, y = true;
4
5      for (int i = 0; i < n - 1; ++i)
6      {
7          if (x && arr[i] > arr[i + 1])
8              x = false;
9          else if (!x && arr[i] < arr[i + 1])
10             y = false;
11      }
12
13     return y;
14 }
```

Какво връща `alg(arr, n)` където `arr` е масив от цели числа с дължина n ? Обосновете отговора си.